REMARKS/ARGUMENTS

This amendment responds to the Office Action of December 13, 2006. Applicant appreciates the detailed examination evident from the Action. Applicant has amended the specification to provide notice to the public that the term "Java" is claimed as a trademark by Sun Microsystems, Inc. Amendments have also been made to the claims as described below. Claims 1-47 remain pending in the application.

I. Objection to the claims - corrected as suggested by the examiner

Dependent claims 17-30 have been corrected, as suggested by the examiner, to indicate that, like their base claim 16, these dependent claims are "method" claims.

II. Original claims 12, 27 and 44 and amended claims 13, 28, and 45 comply with 35 U.S.C. § 112

Claims 12-13, 27-28, and 44-45 stand rejected under 35 U.S.C. § 112, second paragraph for indefiniteness. *Office Action, page 2.* The concern is that these claims contain the name "Java." Ex parte Simpson, 218 USPQ 1020 (Bd App. 1982) is cited for the proposition that where a trademark is used in a claim as a limitation, the claim does not comply with § 112, 2nd paragraph. *Office Action, page 2.* The Examiner notes that a trademark is used "to identify a source" of goods and "not the goods themselves" so that a trademark does not identify or describe the goods. *Office Action, page 3.* Each of these claims refer to a "Java virtual machine" and claims 13, 28, and 45, in particular, refer to a "Java native method interface."

Applicant is aware that Sun Microsystems, Inc. has registered the mark "Java." To dispel any possible confusion by the public, applicant has revised the specification so as to note the proprietary rights claimed by Sun in the Java name. Applicant has further amended claims 13, 28, and 45 to remove the term in connection with "native method interface" as such usage is redundant and unnecessary. As used in connection with "virtual machine," the term "Java" is both valid and appropriate for reasons that will now be discussed.

As a preliminary note, it is not necessarily fatal to the definiteness of an application that a trademark or tradename is used. Rather, it is necessary that "each case" be "decided on its own facts." In re Metcalfe and Lowe, 161 U.S.P.Q.2nd 789, 792 (C.C.P.A. 1969). See also In re Gebauer-Fuelnegg et al., 50 USPQ 125 (C.C.P.A. 1941) where four claims referencing the trademark "pliolite" were held to be valid. In accord are Ex Parte Jerry Kitten (BPAI, heard Sept. 16, 1999, Appeal No. 19960513); and Ex Parte William Z. Goldstein (BPAI, heard June 7, 2005, Appeal No. 20050823). The last two "unpublished" decisions are viewable at http://des.uspto.gov/Foia/BPAIReadingRoom.jsp.

The term "Java virtual machine" had a recognized and particular meaning to those of ordinary skill in the art at the time the instant application was filed. According to the "Microsoft Computer Dictionary," Fourth Edition (Microsoft Press 1999), a Java Virtual Machine is an environment in which Java programs run. More technically, this term signifies a virtual machine that can interpret and execute Java bytecode, such as compiled from a Java application, and that can produce an output "customized" for a "particular platform," such as a host operating system. See Wikipedia, Java Virtual Machine, http://en.wikipedia.org/wiki/Java_virtual_machine. In short, the phrase "Java virtual machine" has an ordinary and customary meaning when examined "through the viewing glass" of a person skilled in the art. Home Diagnostics, Inc. v. Lifescan, Inc., 381 F.3d 1352, 1358, 72 USPQ2d 1276, 1279 (Fed. Cir. 2004). Thus the essential function of §112, second paragraph is fulfilled, which is to provide "adequate notice" to those of ordinary skill in the art of the "metes and bounds" of the invention. In re Goffe, 526 F2d. 1393, 1397, 188 USPQ 131, 135 (CCPA 1975) and In re Hammack, 427 F.2d 1378, 1382, 166 USPQ 204, 208 (CCPA 1970).

The complete specification for creating an official implementation of a Java Virtual Machine has been published in book format, see "Java™ Virtual Machine Specification," 2nd Edition, by Tim Lindholm and Frank Yellin (Prentice Hall 1999). Thus a "Kaffe" implementation of a "Java virtual machine" has been developed by a source working independently of Sun Microsystems in a "clean room" environment (e.g., working backward from only the specified functions of the "Java virtual machine"). See Wikipedia, Kaffe, http://en.wikipedia.org/wiki/Kaffe. There are further implementations from yet other sources. See Wikipedia, List of Java virtual machines,

http://en.wikipedia.or/wiki/List_of_Java_virtual_machines. A Java virtual machine developer may also seek approval or certification from Sun Microsystems that its particular implementation is "compliant" (e.g., deserving of the steaming cup logo) which requires passing an appropriate technology compatibility test. Even here, however, the term "Java" in "Java virtual machine" functions more as a certification mark (i.e., attesting to the product meeting certain performance standards) than as a regular trademark (i.e., indicating a unique source). It might be noted that the term "Java HotSpot" is how Sun refers to their own implementation of the Java virtual machine

Based on the foregoing amendments and remarks, applicant respectfully maintains that claims 12-13, 27-28, and 44-45 fully comply with Section 112, second paragraph.

III. The rejection under 35 U.S.C. § 102 is traversed

Claims 1-9, 16-24 and 31-41 stand rejected under 35 U.S.C. 102(b) for anticipation by Bacon (U.S. Pat. 6,247,025). Applicant has carefully reviewed this reference and the examiner's comments and respectfully traverses this ground of rejection.

As the examiner is more than likely aware, a claim is anticipated only if each and every element of the claimed invention, arranged as in the claim, is found in a single reference. M.P.E.P. § 2131 and Lindemann, Maschinenfabrik GmbH v. American Hoist & Derrick Co., 221 USPQ 481, 485 (Fed. Cir. 1984). The identical invention must be shown in as complete detail as is contained in the claims. MPEP § 2131 and Richardson v. Suzuki Motor Co., 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). Under this standard, it is untenable to say that Bacon anticipates the identified claims.

Subparagraph (a) of each independent claim (claims 1, 16, and 31) require a shared object space storing an object accessible to at least two applications simultaneously (in claim 16, the "simultaneously" limitation may be inferred from the preamble which speaks of "synchronizing access" for plural applications). FIG. 1 of Bacon shows an "application program" 102. Even granting that this "program" can be deemed to include plural applications (col. 4, lines 37-38), Bacon does not specifically teach enabling these applications to simultaneously access the same object in a shared object space. Although Bacon does suggest multithread access, one application may run multiple threads, so that multithread access does not necessarily signify multi-application access. In fact, the conventional approach is to enforce "platform independence" by

having each application using its own (not shared) object space. This is done to prevent conflict between applications for the same object which can result in one failed application causing others to fail as well.

Though the above paragraph targets the anticipation rejection, it is well to note, at this juncture, that McGuire et al. (U.S. Pat. App. 2003/0097360), also cited, does suggest a system in which multiple virtual machines (e.g., plural applications) share a common object space. However, McGuire has its own system for avoiding conflicts using "flat" and "inflated" objects (see figs. 4A & B), and one of ordinary skill would not have been motivated to transplant the locking/unlocking system of Bacon into the already functioning conflict system of McGuire.

In similar vein, subparagraph (b) of each independent claim (1, 16, and 31) calls for lock nodes or locks, where each lock node or lock is uniquely associated with one of plural applications. At best, Bacon shows each lock being associated with only one application.

Subparagraph (c) of independent claims 1 and 16 calls for the sharable object to have an object header that can store any selected one of three types of items: (1) a selective said lock node; (2) a reference to a selective one of the lock nodes stored in the lock table; and (3) a default value other than types (1) or (2) above. Applicant has carefully reviewed the related portions of Bacon cited by the examiner (col. 5, lines 7-67), but has been unable to identify an object header that stores all of these three types.

In Bacon, it is suggested that the lock may be contained "within the object" (col. 5, line 15-17) and may comprise "a thread identifier" (col. 5, lines 18-29). Bacon also suggests "in the alternative" that the lock may be "located in a table outside the object and looked up by hashing" (col. 5, lines 17-19). The first Bacon lock may be deemed to correspond to item 1 above ("a selective said lock node") and the second Bacon lock may be deemed to correspond to item 2 above ("a reference to a selective one of the lock nodes" stored in a lock table). However, teaching that one type of header can include element A (e.g., item 1) and a different "alternative" type of header can include element B (e.g., item 2) is not the same as teaching that a single header can include any selected one of both elements A and B (e.g., any selected one of both items 1 and 2).

Bacon suggests that the lock can include a flag or "Bacon bit" that can take on a "0" value when the object is not locked (col. 5, lines 28-31 and lines 50-53). This would seem to

correspond with item 3 (the "default value") identified above. Again, however, though Bacon may suggest an object header for storing items 1 and 3 above or, in the alternative, items 2 and 3 above, this is not the same as teaching an object header for storing any selected one of all the items 1, 2, and 3 above.

The dependent claims include limitations that further refine the meaning of the three items just discussed. Thus, claims 4 and 19 identify item 1 (the "selective said lock node") as operating as "a cheap lock." Claims 5, 6, 20 and 21 identify item 2 (the "reference to a selective one of said lock nodes stored in said lock table") as "an expensive lock." Here item 2 is more "expensive" than item 1 because the information about which application and thread owns the lock can be directly read from the "cheap lock" of item 1 whereas the lock node storing such information is only referenced in item 2.

The above explains why independent claim 31 speaks of a lock "being either a cheap lock or an expensive lock" In independent claim 31, the cheap lock is defined as a lock inserted into or removed from the header in a relatively short time with respect to the expensive lock (note that inserting or removing the self-contained "cheap" lock, as contained only in the header, requires fewer steps or processing overhead than inserting or removing the "expensive lock," which is associated with both a "reference" in the header and a corresponding "lock node" in the lock table). Bacon does not anticipate claim 31 (in addition to the reasons already stated) because it suggests a header having either a type 1 (e.g., cheap lock) or, in an alternative embodiment, a type 2 (e.g., expensive lock) but not either type of lock flexibly included in the header of the same embodiment.

In rejecting claim 32 (*Office Action, top of page 6*) and claim 40 (*Office Action, top of page 5*), the Examiner appears to equate the "Bacon bit" (by citing col. 5, lines 56-65 of Bacon) or "Global lock" (by citing col. 6, lines 54-61 of Bacon) with applicant's "expensive" lock. Both the "Bacon bit" and "Global lock" are one-bit flags that take on either a "0" or "1." As base claim 31 indicates, an "expensive" lock takes a relatively longer time to insert or remove from a header than a "cheap" lock, but a one-bit flag requires even less time to insert or remove than a multi-bit "cheap" lock. Hence neither a Bacon bit nor a Global lock is an "expensive" lock in accordance with applicant's claims (i.e., see claims 5, 6, 20, 21, 31, 32, and 40). For that matter, neither a

Bacon bit nor a Global lock represents "a reference to" a "lock node stored" in a lock table (i.e., see claims 1 and 16).

There are yet further distinctions to be found between the subject matter of the claims and the mechanism described in Bacon. For example, the rejection of claims 8, 23, and 41 (*Office Action, bottom of page 5*) purports to find a "lock manager" in Bacon, yet after careful review of the cited portions of Bacon (col. 6, lines 6-67; col. 7, lines 65-67, col. 8, lines 1-18), applicant finds no explicit mention of this element. Col. 8, lines 1-18, for example, essentially describe restricting the number of times the Global lock can "spin" to prevent thread lockout. A one-bit global lock or a component for controlling only a global lock is not a "lock manager."

It is interesting to observe that while Bacon seeks to limit excessive "spin" (col. 2, lines 26-30), a thread seeking to obtain the lock from the current owner is set "spinning" while waiting for the Global lock (col. 6, lines 57-61). Bacon is concerned, for example, that the owning thread might suddenly release the lock (and revert the lock to '0') right after the seeking thread has detected the lock (e.g., detected the nonzero lock value) and so allow an "interloper" thread to acquire the lock before the originally seeking thread has even entered the queue. In accordance with applicant's teachings, this same problem is dealt with differently (e.g., without a Global lock and excessive spinning) using the "cheap" and "expensive" locks already discussed. Referring to paragraph 0070 of applicant's corresponding publication 2005/0086661A1, the seeking thread B "updates" the "cheap" lock of the owning thread A to an "expensive" lock (that refers back to a lock node 336 in the lock table that represents, if you will, a lock "request" ticket). If owning thread A tries, soon afterward, to swap out its cheap lock for '0,' the swap won't take (thereby thwarting lock acquisition by any interloper thread). Owning thread A must either hand off the reference to a lock manager (in a first implementation; see paragraph 0073) or go to the lock table itself (in a second implementation) and swap in thread B's other expensive lock (that refers back to a lock node 337 in the lock table containing relevant hold or acquire information, such as thread B's corresponding application and thread). Of course, in the most common situation, where there is only a thread A and no contending thread B (or C), thread A can efficiently insert and remove its cheap lock (see paragraphs 0069 and 0077). This points out one significant advantage, then, in having a single object header able to store both a self-identifying or cheap lock type and "a reference" or expensive lock type.

Although further distinctions may be found between the subject matter of claims 1-9, 16-24 and 31-41 in contrast with Bacon's teachings, the above distinctions should suffice to establish that these claims are in no way anticipated by Bacon. Applicant maintains, then, that these claims stand in condition for allowance.

IV. The rejection under 35 U.S.C. § 103 is traversed

Dependent claims 10-15, 25-30, and 42-47 have been rejected under 35 U.S.C. 103(a) for obviousness over Bacon (U.S. Pat. 6,247,025) in view of McGuire et al. (U.S. Pat. App. 2003/0097360). Applicant would traverse this rejection. Each of these dependent claims includes all the limitations of the corresponding independent claim (claim 1, 16, or 31). Since these independent claims have already been shown to stand in condition for allowance, the dependent claims are likewise allowable, and further discussion of these dependent claims at this time is unnecessary.

V. Conclusion

Applicant has amended the specification to accord due recognition to trademark rights and corrected informalities in the claims in the manner suggested by the Examiner. Applicant has demonstrated by the remarks herein that the claims comply with Section 112 and that each of the independent claims, together with all their dependent claims, patentably define over the applicable art. Accordingly, applicant respectfully requests the withdrawal of all rejections and the allowance of all pending claims, claims 1-47, in due course.

If the examiner believes that a conference with applicant's undersigned representative would help advance this application to issue, he is invited to contact such representative at the telephone number provided below.

Applicant submits that no fees are required for entry of this Amendment. If it is determined that any fees are due, the Commissioner is hereby authorized to charge such fees to Deposit Account No. 03-1550.

Respectfully submitted

CHERNOFF, VILHAUER, MCCLUNG & STENZEL

Dated: March 2, 2007

By _____

Kevin L. Russell
Reg. No. 38,292
1600 ODS Tower
601 SW Second Avenue
Portland, OR 97204
Tel: (503) 227-5631

Certificate Of Mailing

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Mail Stop Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on March 2, 2007.

Dated: March 2, 2007

_____
Kevin L. Russell